Addressing Cold Start in Recommender Systems: A Semi-supervised Co-training Algorithm

Mi Zhang1,2, Jie Tang1, Xuchen Zhang3,2, Xiangyang Xue1,2
1School of Computer Science, Fudan University
2Shanghai Key Laboratory of Intelligent Information Processing
3Department of Computer Science and Technology, Tsinghua University
{mi_zhang,13210240075,xyxue}@fudan.edu.cn, jietang@tsinghua.edu.cn

ABSTRACT
Cold start is one of the most challenging problems in recommender systems. In this paper we tackle the cold-start problem by proposing a context-aware semi-supervised co-training method named C-SEL. Specifically, we use a factorization model to capture fine-grained user-item context. Then, in order to build a model that is able to boost the recommendation performance by leveraging the context, we propose a semi-supervised ensemble learning algorithm. The algorithm constructs different (weak) prediction models using examples with different contexts and then employs the co-training strategy to allow each (weak) prediction model to learn from the other prediction models. The method has several distinguished advantages over the standard recommendation methods for addressing the cold-start problem. First, it defines a fine-grained context that is more accurate for modeling the user-item preference. Second, the method can naturally support supervised learning and semi-supervised learning, which provides a flexible way to incorporate the unlabeled data.

The proposed algorithms are evaluated on two real-world datasets. The experimental results show that with our method the recommendation accuracy is significantly improved compared to the standard algorithms and the cold-start problem is largely alleviated.

Categories and Subject Descriptors
H.4.m [Information Systems Applications]: Miscellaneous

Keywords
Cold-start; Recommendation; Semi-supervised Learning

1. INTRODUCTION
Recommendation plays an important role in many fields and has attracted a lot of research interest. For example, Netflix has released an interesting fact that about 75% of its subscribers watch series from recommendations. In a recommender system such as Netflix and Amazon, users can browse items and choose those items they are interested in, while the system would also recommend to them the items that the system thought best match their preferences. Afterward, the user may provide feedback (such as rating, usually represented as a score between, for example, 1 and 5) on how the user thinks about an item after she/he has experienced the item. One important task for the recommendation engine is to understand users’ personalized preferences from their historic rating behaviors.

Another important, and actually more challenging task is how to improve the recommendation accuracy for the new (or rarely rated) items and the new (or inactive) users. Comparing to the popular items, for the newly released ones and the old items that are rarely rated by users, it is difficult for the standard recommendation approaches such as collaborative filtering approach to provide high-quality recommendations. Figure 1 shows some preliminary results in our experiments. The recommendation error (by mean-square error, i.e., RMSE) increases quickly with the decrease of popularity of the item. The average error of the most unpopular items (Bin10) almost doubles that of the popular items (Bin1, Cf. Table 2 for details). The problem also exists for the newly entered users or the inactive users who have not contributed enough ratings. Technically, this problem is referred to as cold start. It is prevalent in almost all recommender systems, and most existing approaches suffer from it [22].

Despite that much research has been conducted in this field, the cold-start problem is far from solved. Schein [22] proposed a method by combining content and collaborative data under a single probabilistic framework. However, their method is based on Bayes...
Solution and Contributions. In this paper, we aim to conduct a systematical investigation to answer the above questions. In order to capture users’ preferences, a fine-grained context-aware approach is proposed which incorporates additional sources of information about the users and items rather than users’ rating information only. Furthermore, we propose a semi-supervised co-training method to build the recommendation model. The model not only is able to make use of unlabeled data to help learn the recommendation model, but also has the capacity to build different sub models based on different views (contexts). The built sub models are then combined by an ensemble method. This significantly alleviates the cold-start problem.

We evaluate the proposed model on a publicly available dataset, MovieLens. The results clearly demonstrate that our proposed method significantly outperforms alternative methods in solving the cold-start problem. The overall RSME is reduced by 3-4%, and for the cold-start users and items the prediction accuracy is increased quickly when directly applying the standard approach to unpopular items.

Let $\mathcal{I}$ be the set of items and $\mathcal{U}$ be the set of users in the system. We use $r_{ui}$ to denote a rating that user $u \in \mathcal{U}$ gives to the item $i \in \mathcal{I}$, and let $L = \{r_{ui}\} \subset \mathcal{I} \times \mathcal{U}$ denote all the unknown ratings, i.e., $\mathcal{L} = \mathcal{I} \times \mathcal{U} - L$. We use the notation $|\cdot|$ to denote the cardinality of a set, for example $|L_r|$ indicates the number of ratings in the set $L_r$.

In the following, for an item $i \in \mathcal{I}$ let $d_i$ be $i$’s popularity, which is obtained by $d_i = \frac{\sum u \in \mathcal{U} r_{ui}}{|\mathcal{U}|}$, where $n_i$ is the number of users who have rated $i$ and $m_i$ is the number of users in the recommender system. For a user $u \in \mathcal{U}$, let $d_u$ be $u$’s popularity, which is obtained by $d_u = \frac{\sum i \in \mathcal{I} r_{ui}}{|\mathcal{I}|}$, where $n_u$ is the number of items rated by $u$ and $n$ is the total number of items.

2.1 Approach Overview

To deal with a general recommendation problem, we can consider the standard factorization algorithm [15] as a baseline model. It is a model-based collaborative filtering (CF) algorithm proposed by the winning team in the Netflix competition\footnote{http://www.netflixprize.com/leaderboard}, which is designed for improving the prediction accuracy of the Netflix movie recommender system. This algorithm can be regarded as a state-of-the-art algorithm. The model is defined as

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

Here, the observed rating is factorized into four components: global average $\mu$, user bias $b_u$, item bias $b_i$, and user-item interaction $q^T p_u$ that captures the user’s personalize preference on item $i$. By minimizing the difference of the real ratings and the factorized ratings in the labeled training data, we can estimate the unknown parameters $\{b_u\}, \{b_i\}, \{q_i\}, \{p_u\}$ in Eq. 1. This model is referred to as FactCF in the rest of the paper.

However, the standard CF approach cannot deal with the cold-start problem. As shown in Figure 1, the recommendation error increases quickly when directly applying the standard approach to unpopular items.

In this paper, we propose a novel semi-supervised co-training approach, i.e., CSEL, to address the cold-start problem. Specifically, at the high-level, the approach consists of two stages:

- **Context-aware factorization.** We propose a fine-grained context model by adding in the contexts of users and items, specifically, the interactions between them into the model. The model is the basis to deal with the cold-start problem;
- **Semi-supervised Co-training.** Based on the context modeling results, a semi-supervised learning framework is built for addressing the cold-start problem in recommender systems, where multiple models are constructed using different methods.
information and then co-training is applied so that the learned models can help (teach) each other. This is the major technical contribution of this work. Besides the specific models used in this work, the proposed framework is very flexible and can accommodate any kind of models without any extra information.

3. CONTEXT-AWARE FACTORIZATION

Often in a recommender system, many users supply very few ratings, making it difficult to reach general conclusions on their taste. One way to alleviate this problem is to incorporate additional sources of information, for example, recommender systems can use the attributes of the users and items to build a context of the user preference. Following this thread, we propose a context-aware factorization model by leveraging “more general” sources such as age and gender of a user, or the genres and tags of an item that are independent of ratings. Saying that they are “more general” than the ratings is in a sense that a rating is given by a specific user to a specific item, whereas these sources can be learned from all the users or items that share the same category. In this section, based on the FactCF algorithm given in Eq. 1 we propose a context-aware model that incorporates the context information into the model.

Here we start enhancing the model by letting item biases share components for items linked by the genres. For example, items in a certain genre may be rated somewhat higher than the average. For each type of learned parameter we set a distinct learning rate such that, e.g., parameters that appear more often in a category, are the parts mixing the contexts. For example, items in L, rating (in

\[ r_{ui} = \mu + b_u + b_i + q^T p_u + \frac{1}{|\text{genres}(i)|} \sum_{g \in \text{genres}(i)} b_g, \] (2)

then the total bias associated with an item i sums both its own specific bias modifier \( b_i \) together with the mean bias associated with its genres \( \frac{1}{|\text{genres}(i)|} \sum_{g \in \text{genres}(i)} b_g \). One could view these extensions as a gradual accumulation of the biases. For example, when modeling the bias of i, the start point is

\[ \sum_{g \in \text{genres}(i)} b_g, \] and then \( b_i \) adds a residual correction on the top of this start point. A similar method has been also used for music recommendations [7].

The second type of “general” information we incorporate into the model is the context (attributes) of users such as age and gender. For example, we can enhance Eq. 2 as follows:

\[ \hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u + \frac{1}{|\text{genres}(i)|} \sum_{g \in \text{genres}(i)} b_g, \] (3)

where \( b_u, b_i \), and \( b_g \) are the biases associated with the user’s age, occupation and gender, respectively. Moreover, we propose mixing the user and item’s contexts as below to obtain a further optimization:

\[ \hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u + \frac{1}{|\text{genres}(i)|} \sum_{g \in \text{genres}(i)} b_{ug}, \] (4)

where \( b_{ug}, b_u, b_i, o \) and \( b_g \) are the parts mixing the contexts. Specifically, \( b_{ug} \) is user u’s bias in genre g, i.e., \( b_{ug} \) catches u’s preference in g. We can use a stochastic gradient descent (SGD) algorithm to learn the parameters. Compared to \( b_i \) that is independent of specific users, here \( b_{ug} \) is only updated when the current rating (in L) used for training is given by u. Likewise, \( b_u, b_i, o \) and \( b_{ug} \) are item i’s preferences by the users in the categories of a, o and s, respectively, and they’ll be updated in the learning process when the rating is associated with i and the users are in the corresponding categories.

Model Learning. By combining all the context information, we can define an objective function to learn the context model by minimizing the prediction errors over all the examples in L.

\[ \min_{p_u, q_u, b_u} \sum_{(u,i) \in R} (r_{ui} - \hat{r}_{ui})^2 + \lambda \|p_u\|^2 + \|q_i\|^2 + \sum \beta^2 \] (5)

where \( r_{ui} \) can be any model defined in Eqs. 2-4; \( \| \cdot \| \) indicate the 2-norm of the model parameters and \( \lambda \) is the regularization rate. The objective function can be solved by the stochastic gradient descent (SGD), SGD processes the training examples one-by-one, and update the model parameters corresponding to each example. More specifically, for training example \( r_{ui} \), SGD lowers the squared prediction error \( e_{ui}^2 = (r_{ui} - \hat{r}_{ui})^2 \) by updating each individual parameter \( \theta \) by

\[ \Delta \theta = -\gamma \frac{\partial e_{ui}^2}{\partial \theta} - \lambda \theta = 2\gamma e_{ui} \frac{\partial \hat{r}_{ui}}{\partial \theta} - \lambda \theta \]

where \( \gamma \) is the learning rate. Thus, the parameters are updated by moving in the opposite direction of the gradient, yielding:

\[ q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i) \]
\[ p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u) \]
\[ b_u \leftarrow b_u + \gamma \cdot (e_{ui} - \lambda \cdot b_u) \]

For each type of learned parameter we set a distinct learning rate and regularization rate. This grants us the flexibility to tune learning rates such that, e.g., parameters that appear more often in a model are learned more slowly (and thus more accurately). Similarly, the various regularization coefficients allow assuming different scales for different types of parameters.

The model given by Eq. 4 turns out to provide a better performance than the other ones in terms of prediction accuracy therefore it is adopted in this work for the further optimization with semi-supervised co-training.

4. SEMI-SUPERVISED CO-TRAINING

Based on the learned contexts, we propose a semi-supervised co-training (CSEL) framework to deal with the cold-start problem in recommender systems. CSEL aims to build a semi-supervised co-training model are learned more slowly (and thus more accurately). Similar-

\[ \min_{p_u, q_u, b_u} \sum_{(u,i) \in R} (r_{ui} - \hat{r}_{ui})^2 + \lambda \|p_u\|^2 + \|q_i\|^2 + \sum \beta^2 \] (5)

where \( r_{ui} \) can be any model defined in Eqs. 2-4; \( \| \cdot \| \) indicate the 2-norm of the model parameters and \( \lambda \) is the regularization rate. The objective function can be solved by the stochastic gradient descent (SGD), SGD processes the training examples one-by-one, and update the model parameters corresponding to each example. More specifically, for training example \( r_{ui} \), SGD lowers the squared prediction error \( e_{ui}^2 = (r_{ui} - \hat{r}_{ui})^2 \) by updating each individual parameter \( \theta \) by

\[ \Delta \theta = -\gamma \frac{\partial e_{ui}^2}{\partial \theta} - \lambda \theta = 2\gamma e_{ui} \frac{\partial \hat{r}_{ui}}{\partial \theta} - \lambda \theta \]

where \( \gamma \) is the learning rate. Thus, the parameters are updated by moving in the opposite direction of the gradient, yielding:

\[ q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i) \]
\[ p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u) \]
\[ b_u \leftarrow b_u + \gamma \cdot (e_{ui} - \lambda \cdot b_u) \]

For each type of learned parameter we set a distinct learning rate and regularization rate. This grants us the flexibility to tune learning rates such that, e.g., parameters that appear more often in a model are learned more slowly (and thus more accurately). Similarly, the various regularization coefficients allow assuming different scales for different types of parameters.

The model given by Eq. 4 turns out to provide a better performance than the other ones in terms of prediction accuracy therefore it is adopted in this work for the further optimization with semi-supervised co-training.

4. SEMI-SUPERVISED CO-TRAINING

Based on the learned contexts, we propose a semi-supervised co-training (CSEL) framework to deal with the cold-start problem in recommender systems. CSEL aims to build a semi-supervised learning process by assembling two models generated with the above context-aware model, in order to provide more accurate predictions. Specifically, CSEL consists of three major steps.

- **Constructing multiple regressors.** Since we are addressing a regression problem, the first step is to construct the two regressors, i.e. \( h_1 \) and \( h_2 \) from L, each of which is then refined with the unlabeled examples that are labeled by the latest version of their peer regressors.

- **Co-Training.** In the co-training, each regressor can learn from each other. More accurately, those examples with high confidence are selected for the regressor and being labeled (predicted) by it, and later to be used to “teach” the other regressors.

- **Assembling the results.** In the end the results obtained by the individual regressors are assembled to form the final solution.
Generating the regressors could be the standard factorization model collaboratively in the co-training process. The basic learner for from the original training set, and two regressors are trained on is called a Constructing Regressors by Manipulating the Training Set. Adopted in this paper to generate multiple regressors for co-training. By manipulating the training set, while the other way is to build up more regressors, constructing the teaching sets, and assembling. Note more details will be given about the methods of constructing multiple views (more than two). In this work, for the sake of simplicity we focus on two regressors.

4.1 Constructing Multiple Regressors
Generally speaking, in order to generate different learners for ensemble, one way is to train the models with different examples by manipulating the training set, while the other way is to build up different views by manipulating the attributes [6], which are both adopted in this paper to generate multiple regressors for co-training.

Constructing Regressors by Manipulating the Training Set. A straightforward way of manipulating the training set is Bagging. In each run, Bagging presents the learning algorithm with a training set that consists of a sample of $k$ training examples drawn randomly with replacement from the original training set. Such a training set is called a bootstrap replicate of the original training set.

In this work two subsets are generated with the Bagging method from the original training set, and two regressors are trained on the two different subsets, respectively, for the purpose of working collaboratively in the co-training process. The basic learner for generating the regressors could be the standard factorization model (Eq. 4), i.e.,

$$h_1(u, i) = h_2(u, i) = \mu + b_u + b_i + q_i^T p_u + \sum_{g \in \text{genres}(i)} \frac{b_{ug}}{|\text{genres}(i)|} + b_{u0} + b_{i0} + b_{s}$$  

Some related work tries to use diverse regressors to reduce the negative influence of the newly labeled noisy data [30]. Our experiments also showed that a good diversity between the regressors indeed helps much in improving the performance of co-training. Thus it is an important condition for generating good combinations of regressors. Intuitively the diversity between the two regressors generated in this way are coming from the different examples they are trained on, and can be evaluated by their difference on predictions. Moreover, another important condition to make a good ensemble is, the two regressors should be sufficient and redundant. Then being translated to this case, it means each training set should be sufficient for learning, respectively and the predictions made by the two individual regressors should be as accurate as possible.

However, the above two conditions, diversity and sufficiency, are contradictory to each other. That is, let the size of the two subsets generated with the Bagging method both be $k$, then with the original training set size fixed, when $k$ is big, the overlap between the two subsets will be big as well, which will result in a small diversity between regressors. In the mean time the accuracy performance of individual regressors will be good with a big set of training examples. And vice versa when $k$ is small. Thus in the experiments we need to find an appropriate value of $k$ for the trade-off between these two criteria. This method is referred to as $M_1$, hereafter.

Constructing Regressors by Manipulating the Attributes. Another way to generate multiple regressors is to divide attributes into multiple views, such as

$$h_1(u, i) = \mu + b_u + b_i + q_i^T p_u + \frac{1}{|\text{genres}(i)|} \sum_{g \in \text{genres}(i)} b_{ug}$$  

$$h_2(u, i) = \mu + b_u + b_i + q_i^T p_u + b_{i0} + b_{i0} + b_{s}$$  

The two regressors are both a part of the model given in Eq. 4. Unlike $M_1$, here the regressors are both trained on the entire training set. To guarantee enough diversity between the constructed regressors, the contexts are separated, specifically, user-related contexts and item-related contexts are given to the different regressors. Further, in order to keep a good performance in terms of prediction accuracy (with respect to the sufficient condition) for the individual regressors, the common part, $\mu + b_u + b_i + q_i^T p_u$, is kept for both regressors. This method is referred to as $M_2$, hereafter.

Constructing Regressors by a Hybrid Method. As described above, $M_2$ is to train a same model on different subsets whereas $M_1$ is to construct different models and let them be trained on a single set. The hybrid method is to construct different regressors as well as training them on the different subsets of the attributes. Obviously, it will bring in more diversity between the regressors. The hybrid method is referred to as $M_{hyd}$.

4.2 Semi-supervised Co-training
Now with the multiple regressors at hand, the task becomes how to co-train the different regressors. Specifically, we need to construct a “teaching set” $T_j$ for each regressor $j$ and use it to teach its peer (the other regressors). This is a key step for launching the semi-supervised learning (SSL). One challenge here is how to determine the criteria for selecting unlabeled examples from $U'$ so as to build the teaching set. The criteria we used in this work is the regressor’s confidence. Every candidate example has a specific confidence value to reflect the probability of its prediction to be accurate. As given in the algorithm, before SSL processes a smaller set $U' \subset U$ is drawn randomly from $U$ because $U$ could be huge.

Confidence for the FactCF Model. In many cases the user can benefit from observing the confidence scores [11], e.g. when the system reports a low confidence in a recommended item, the user may tend to further research the item before making a decision. However, not every regression algorithm can generate a confidence
by itself, like the factorization regressor this work is based on. Thus it is needed to design a confidence for it.

In [30] the authors proposed the predictive confidence estimation for the kNN regressors. The idea is that the most confidently labeled example of a regressor should decrease most the error of the regressor on the labeled example set, if it is utilized. However, it requires the example to be merged with the labeled data and re-train the model, and this process needs to be repeated for all candidate examples. Similarly, in [10] another confidence measure was proposed, again, for the kNN regressor. The idea is to calculate the conflicts level between the neighbors of the current item for the current user. However, the above two methods are both based on the k nearest neighbors of the current user and cannot be directly applied to our model. In the following we define a new confidence measure that suits our model.

Confidence in the recommendation can be defined as the system’s trust in its recommendations or predictions [11]. As we have noted above, collaborative filtering recommenders tend to improve their accuracy as the amount of data over items/users grows. In fact, how well the factorization regressor works directly depends on the number of labeled examples (ratings) that is associated with each factor in the model. And the factors in the model pile up together to have an aggregate impact on the final results. Based on this idea, we propose a simple but effective definition for the confidence on a prediction made by regressor $j$ for example $x_{ui}$:

$$C_j(x_{ui}) = \frac{d_j^{(1)} \times d_j^{(2)} \times \prod_{d \in G,O,A,S} d_i^{(j)}}{\mathcal{N}}$$

(9)

where $\mathcal{N}$ is the normalization term; $d_j^{(1)}$ and $d_j^{(2)}$ are the user $u$ and item $i$’s popularities associated with regressor $j$.

The idea is, the more chances (times) a factor gets to be updated during the training process, the more confident it is with the predictions being made with it. The rating number that is associated with the user-related factors $b_u$ and $p_u$ in the model is expressed by $d_u$, while the rating number that is associated with the item-related factors $b_i$ and $g_i$ is expressed by $d_i$. Here the assumption is that the effects of the parameters are accumulative.

Confidence for the Context-aware Models. With the similar idea the confidence for the context-aware models is defined as below:

$$C_j(x_{ui}) = \frac{d_j^{(1)} \times d_j^{(2)} \times \prod_{d \in G,O,A,S} d_i^{(j)}}{\mathcal{N}}$$

(10)

where $c$ stands for the general information including genre, age, gender, occupation, etc.; $d_i$ represents the fraction of items that fall in a certain category $c$. For example, $d_{g_1}$ is the fraction of items that fall in the genre “Fictions”; $d_{g_2}$ is the fraction of items that fall in the genre “Actions”; etc., with $G = \{g_1, ..., g_{|G|}\}$ (there are 18 genres in the MovieLens dataset). Likewise, $d_a$, $d_s$, $d_o$, are associated with the categories of $A = \{a_1, ..., a_{|A|}\}$ (the entire range of age is divided into 7 intervals), $S = \{male, female\}$ and $O = \{o_1, ..., o_{|O|}\}$ (20 kinds of occupations), respectively, can be obtained in the same way.

Specifically, for the model with the item-view, i.e., $h_1$ in Eq. 7, the confidence for $x_{ui}$ is $C_1(x_{ui}) = d_1^{(1)} \times d_1^{(2)} \times d_a$, where $d_a = \frac{1}{\sum_{g \in genres(i)} \sum_{o \in occupations(i)} d_{go}}$ and $\{genres(i)\}$ is the number of genres $i$ belongs to. On the other hand, for the model with the user-view, i.e., $h_2$ in Eq. 8, the confidence is $C_2(x_{ui}) = d_2^{(2)} \times d_1^{(2)} \times d_a \times d_s \times d_o$. This way the confidence depends on the view applied by the model.

With this definition, the two regressors (given by Eq. 6) being trained on different sample sets ($M_i$) will naturally have different $d_u$, $d_i$ and $d_o$. E.g., for a user $u$, its $d_u^{(1)}$ for $h_1$ is the rating numbers of $u$ being divided into the first sub-set, while its $d_u^{(2)}$ for $h_2$ is the rating numbers of $u$ being divided into the second sub-set, and $d_i$ likewise. Therefore, it will result in two different confidence values for these two models.

Note that in this work the confidence value of a prediction, $C_j(x_{ui})$, is for selecting examples within a single model $j$, in order to steer the process of semi-supervised learning. Hence they are not comparable between different models. For the same reason the normalization part $\mathcal{N}$ is a constant within a model and thus can be omitted.

**Constructing and Co-training with the Teaching Set.** For each regressor $j$, when constructing the “teaching set” $T_j$ with $C_j(x_{ui})$, to avoid focusing on the examples that are associated with the most popular users and items that have relatively high $d_u^{(1)}$ and $d_i^{(1)}$, rather than directly selecting the examples with the highest confidence, we obtain a probability for each candidate example based on $C_j(x_{ui})$ and select with the Roulette algorithm [1]. Specifically, the probability given to an example $x_{ui}$ is calculated by:

$$Pr(x_{ui}, j) = \frac{C_j(x_{ui})}{\sum_{x_{ui} \in T_j} C_j(x_{ui})}$$

(11)

What’s noteworthy is, in the experiments we found that simply selecting examples with $Pr(x_{ui}, j)$ sometimes cannot guarantee the performance. Moreover, we observed that a big diversity between the outputs of the two regressors is important for the optimization. The experiments show that for the selected examples, if the difference between the predictions made for them is not big enough the teaching effect can be trivial. This is consistent with the conclusion in [20] where the authors emphasize the diversity between the two learners are important. To guarantee a progress in SSL we propose to apply an extra condition in the teaching set selection. That is, only those examples with a big difference between the predictions made by the two regressors are selected. Specifically, when an example $x_{ui}$ is drawn from $\hat{U}$ with $Pr(x_{ui}, j)$, its predictions made by the two regressors are compared to a threshold $\tau$. If the difference between them is bigger than $\tau$ then $x_{ui}$ is put into $T_j$, otherwise it is discarded.

Next, the examples in the teaching set are labeled with the predictions made by the current regressor, likewise for its peer regressor. Note again that for the sake of diversity the teaching sets for the two regressors should be exclusive.

At last the regressors are updated (re-trained) with these new labeled data. In this way the unlabeled data are incorporated into the learning process. According to the results given in the experiments, by incorporating the unlabeled data the CSEL strategy provides a significant boost on the system performance. This process of constructing the teaching sets and co-training will be repeated for $t$ iterations in the algorithm.

### 4.3 Assembling the Results

The final step of the algorithm is to assemble the results of the regressors after the optimization by SSL. A straightforward method is to take the average, i.e.,

$$h(u, i) = \frac{1}{I} \sum_{j=1}^{I} h_j(u, i)$$

(12)

where $I$ is the number of regressors being assembled. In our case, to assemble results from two regressors it is $h(u, i) = \frac{1}{2} [h_1(u, i) + h_2(u, i)]$.
new methods to tackle the cold-start problem in different applications such as [17, 19, 24, 29]. Work [19] tries to address the cold-start problem as a ranking task by proposing a pairwise preference regression model and thus minimizing the distance between the real rank of the items and the estimated one for each user. Work [29] follows the idea of progressively querying user responses through an initial interview process. [24] is based on an interview process, too, by proposing an algorithm that learns to conduct the interview process guided by a decision tree with multiple questions at each split. [17] aims to recommend apps to the Twitter users by applying latent Dirichlet allocation to generate latent groups based on the users’ followers, in order to overcome the difficulty of cold-start app recommendation. Unlike the above work that addresses the cold-start problem in some specific applications with specific extra sources, our work possesses more generosity. The framework for the CSEL algorithm proposed in this work can basically accommodate any kind of regressors without any extra information required.

The semi-supervised learning algorithm adopted in this paper falls in the category of disagreement-based SSL [31]. This line of research started from Blum and Mitchell’s work [3]. Disagreement-based semi-supervised learning is an interesting paradigm, where multiple learners are trained for the task and the disagreements among the learners are exploited during the semi-supervised learning process. However, little work has been done making use of the semi-supervised techniques in the literature of recommender systems although it is a natural way to solve these problems. Following this line of research we propose a semi-supervised learning algorithm specifically designed for the factorization regressors that outperforms the standard algorithm in both the overall system performance and providing high quality recommendations for the users and items that suffer from the cold-start problem.

6. EVALUATION

6.1 Experimental Setup

Datasets. We evaluate our methods on a public available dataset, MovieLens, consisting of 100,000 ratings (1-5) from 943 users on 1,682 movies. In the experiments, the algorithms are performed at two stages of the system. The first stage is the early stage, i.e., when the system has run for 3 month. 50,000 ratings are collected given by 489 users to 1,466 items. The second stage is the late stage with the entire dataset collected during 7 months. The datasets at these two stages are referred to as $D_1$ and $D_2$, respectively.

The proposed methods are also evaluated on a different version of MovieLens dataset consisting of 1,000,000 ratings provided by 6,040 users for 3,900 movies. The experiments are performed on two stages of it as well. The first stage is when the system has run for 3 months, by this time 200,000 ratings are collected for 3,266 items given by 1,429 users. The second stage is again the full dataset collected in 34 months. They are referred to as $D'_1$ and $D'_2$ for this larger version MovieLens, respectively. The reason we did not do the evaluations on the Netflix dataset is, it only provides the ratings without any attributes or context information about users or items, which makes it impossible to perform the context-aware algorithm and the CSEL method by manipulating the attributes ($\mathcal{M}_r$) proposed in this work.

Besides the historical ratings MovieLens datasets also provides some context information, including 19/18 genres associated with items (movies) and age, occupation and gender associated with users. Gender is denoted by “M” for male and “F” for female, age is given from 7 ranges and there are 20 types of occupations.
In the evaluation, the available ratings were split into train, validation and test sets such that 10\% ratings of each user were placed in the test set, another 10\% were used in the validation set, and the rest were placed in the training set. The validation dataset is used for early termination and for setting meta-parameters, then we fixed the meta-parameters and re-built our final model using both the train and validation sets. The results on the test set are reported.

**Comparison Algorithms.** We compare the following algorithms.

- FactCF: the factorization-based collaborative filtering algorithm given by Eq. 1 [15];
- UB k-NN and IB k-NN: the user-based and item-based k-nearest neighbor collaborative filtering algorithms;
- Context: the context-aware model we propose in Section 3, given by Eq. 4;
- Ensemble, and Ensemble*: directly assembling the results of $h_1$ and $h_2$ generated by manipulating the training set ($M_s$) and manipulating the attributes ($M_v$ in Section 4.1), respectively. Among the options of ensemble method in Section 4.3, the weighted method in Eq. 13 gives a slightly better result than the other ones and thus being adopted.
- CSEL, CSEL*, and CSEL*: the semi-supervised co-training algorithms using the $h_1$ and $h_2$ generated by $M_s$, $M_v$, and $M_v$ (cf. Section 4.1), respectively.

Besides the state-of-the-art FactCF algorithm we also use another standard algorithm, k-NN, to serve as a baseline. Two kinds of k-NN algorithms are implemented and compared to our methods. With user-based k-NN, to provide a prediction for $x_{ui}$, the $k$ nearest neighbors of $u$ who have also rated $i$ are selected according to their similarities to $u$, and the prediction is made by taking the weighted average of their ratings. With item-based k-NN, the $k$ nearest neighbors of $i$ that have also been rated $u$ are selected to form the prediction. Note that as another standard recommendation approach, the content-based algorithm has a different task from ours and thus not considered in this work. It aims to provide top-N recommendations with the best content similarities to the user, rather than providing predictions.

When training the models we mainly tuned the parameters manually and sometimes resorted to an automatic parameter tuner (APT) to find the best constants (learning rates, regularization, and log basis). Specifically, we were using APT2, which is described in [27]. All the algorithms are performed with 10-fold cross-validations.

**Evaluation Measures.** The quality of the results is measured by the root mean squared error of the predictions:

$$RMSE = \sqrt{\frac{\sum_{(u,i) \in TestSet} (r_{ui} - \hat{r}_{ui})^2}{|TestSet|}}$$

where $r_{ui}$ is the real rating and $\hat{r}_{ui}$ is the rating estimated by a recommendation model.

### 6.2 Results

The overall performance on the four datasets is given in Table 1. The CSEL methods outperform both k-NN and FactCF algorithms in terms of the overall RMSE. Specifically, compared to FactCF the prediction accuracy averaged on all the test examples is improved by 3.4\% on $D_1$, 3.6\% on $D_2$, 3.2\% on $D_1'$ and 3.3\% on $D_2'$ respectively, with CSEL* performing the best among the methods. An unexpected result is that the hybrid method CSEL* does not outperform CSEL. It is because the double splitting, i.e., on both the training set and views, leads to a bigger diversity (see Table 4) as well as worse accuracy for the regressors. As expected, FactCF outperforms k-NN on the overall performance by exploiting the latent factors. We tried different values of $k$ from 10 to 100, $k = 20$ gives the best results. CSEL presents a much bigger advantage over the k-NN algorithms, i.e., 12\% over UB and 16\% over IB. The optimal solution is obtained by the CSEL** for all datasets.

**Cold-start Performance.** Except for the overall RMSE performance, we also present the recommendation performance according to the popularity of items. Specifically, we estimate the popularity of each item based on the number of ratings and partition all the items into 10 bins according to their popularity with equal number, the average RMSE scores are obtained for each bin. The results on these 10 bins over dataset $D_2$ are depicted in Table 2 corresponding to $Bin_i^{(1)}$ through $Bin_i^{(10)}$. Likewise, all the users are partitioned into 10 bins, the average RMSE scores are reported for each bin, corresponding to $Bin_u^{(1)}$ through $Bin_u^{(10)}$ in the table. The average rating number in each bin is given, too. With the users and items being grouped by their popularity we can observe how well the cold-start problem is solved. For the presented results, the differences between the algorithms are statistically significant ($p < 0.05$ with t-test). Due to space limitation, we do not list results on all the datasets, as according to the experimental results the behaviors of the algorithms on the four datasets ($D_1$, $D_2$, $D_1'$, and $D_2'$) turned out to be similar with each other. Table 2 demonstrates how well the cold-start problem can be addressed by the proposed CSEL algorithm. The performance of the k-NN algorithms shows again that the cold-start problem is common to all kinds of recommendation algorithms, i.e., the items and users with less ratings receive less accurate predictions. The reason is that the effectiveness of the k-NN collaborative filtering recommendation algorithms depends on the availability of sets of user-similar items with similar preferences [12]. Compared to the baseline algorithms, CSEL successfully tackles the cold-start problem by largely improving the performance of the unpopular bins. Again, CSEL* provides the best results, compared to FactCF, RMSE drops up to 8.3\% for the most unpopular user bins and 10.1\% for the most unpopular item bins. Compared to UB k-NN, it drops up to 13.5\% for the most unpopular user bins and 22.3\% for the most unpopular item bins, respectively.

| Table 1: The overall RMSE Performance on the Four Datasets. |
|-----------------|-----------------|-----------------|-----------------|-----------------|
| Models          | $D_1$           | $D_2$           | $D_1'$          | $D_2'$          |
| UB k-NN         | 1.0250          | 1.0267          | 1.0224          | 0.9522          |
| IB k-NN         | 1.0733          | 1.0756          | 1.0709          | 1.0143          |
| FactCF          | 0.9310          | 0.9300          | 0.9260          | 0.8590          |
| Context         | 0.9164          | 0.9180          | 0.9140          | 0.8500          |
| Ensemble        | 0.9174          | 0.9170          | 0.9156          | 0.8490          |
| Ensemble*       | 0.9131          | 0.9153          | 0.9130          | 0.8452          |
| CSEL            | 0.9013          | 0.9061          | 0.9022          | 0.8375          |
| CSEL*           | 0.8987          | 0.8966          | 0.8963          | 0.8334          |
| CSEL**          | 0.9012          | 0.9020          | 0.9000          | 0.8355          |

**6.3 Analysis and Discussions**

We further evaluate how the different components (context-aware factorization, ensemble methods, etc.) contribute in the proposed algorithm framework. In the following the performance of the proposed context-aware model and ensemble methods are given, and the results of CSEL are further analyzed.
between 18 genres but not exclusively, which leaves relatively richer rating, the ratings for each item are divided by 2 for $S$; $O; A$ related to the specific $i$ the average rating number of users is 106. The category. Specifically, the average rating number of items is 60 and be traced back again to the rating numbers falling into into each more than $b$. As depicted in Table 2. There is a certain level improvement on all the compared to FactCF. The performance on the cold-start problem is maintained by two approaches $M_1$ and $M_2$ to form the teaching set is essential to the success of the SSL process. In order to do so, a threshold $\beta$ is set to select the examples with the top 10% biggest difference. So, according to the distribution of the difference depicted in Figure 2 we have $\beta = 0.675$ for $M_1$, $0.85$ for $M_2$, and 1 for $M_{CSEL}$, respectively. As expected, $M_{CSEL}$ generates the most diverse regressors whereas $M_{CSEL}$ generates the least. During the iterations the difference $\times|b|_i$ for $M_{CSEL}$ and $M_{CSEL}$ are set to 0.8, i.e., $0.8 \times |\tau|$ examples are randomly selected for each generate the bootstrap duplicates with the $M_{CSEL}$ algorithm. Given the individual regressors $h_{\nu_1}$ and $h_{\nu_2}$’s performance in Table 3, by simply assembling them together before any optimization with SSL, the accuracy has been largely improved. Table 2 shows the performance on bins. Comparing to Context, with a slightly better performance on the overall RMSE, their performance on the unpopular bins are better. The biggest improvement is observed in the last bin, where the RMSE is reduced by 4.6% for users and 6% for items. The empirical results demonstrate that the ensemble methods can be used to address the cold-start problem. For a more theoretical analysis for the ensemble methods, please refer to [6].

**Semi-supervised Co-training Algorithm (CSEL).** In order to generate the bootstrap duplicates with the $M_{CSEL}$ method, we set a parameter $0 \leq \beta \leq 1$ as the proportion to take from the original training set. In our evaluation, by trying different values $\beta$ is set to 0.8, i.e., $0.8 \times |\tau|$ examples are randomly selected for each duplicate, which provides the best performance in most cases. As mentioned in Section 4.2, selecting the examples with a big difference between $h_1$ and $h_2$ to form the teaching set is essential to the success of the SSL process. In order to do so, a threshold $\beta$ is set to select the examples with the top 10% biggest difference. So, according to the distribution of the difference depicted in Figure 2 we have $\beta = 0.675$ for $M_1$, $0.85$ for $M_2$, and 1 for $M_{CSEL}$, respectively. As expected, $M_{CSEL}$ generates the most diverse regressors whereas $M_{CSEL}$ generates the least. During the iterations the difference changes with the updated regressors. That is, if there is not enough unlabeled examples meet the condition the threshold will automatically shrink by $\beta = 0.8 \times \beta$. The performance on solving the cold-start problem on $D_2$ is depicted by the comparison of the left and right figures in Figure 3, which presents the results on user bins of the FactCF method and the $M_{CSEL}$ method, respectively. The
We tried many possible values of $t$ and re-trained the individual models for different times of the standard factorization algorithm since the cost of the algorithm is $t$ times of the standard factorization. Meanwhile, the performance on the unpopular bins has also been improved, which means the overall RMSE performance is generally improved. The improvement on the unpopular bins means the cold-start problem is well addressed in this respect to obtain an even better initial point at the beginning of the SSL process.

**Efficiency Performance.** Roughly speaking, the cost of the algorithm is $t$ times of the standard factorization algorithm since the individual models are re-trained for $t$ times. In the experiments we found that teaching set size $N$ has a big influence on convergence. We tried many possible values of $N$ from 1,000 up to 20,000 with respect to the optimization effect and time efficiency, and found that 20% of the training set size is a good choice, e.g., $N = 8,000$ for $D_1$ and $N = 16,000$ for $D_2$, which leads to a fast convergence and optimal final results. Besides, we found the quality of the teaching set, i.e., difference and accuracy of the predictions made for the selected examples, has an even bigger impact on the performance and convergence of CSEL. With the methods described in Section 4.2 we managed to guarantee the best examples that satisfy the conditions are selected. The convergence point is set to when the RMSE in the validation set drops any more for 2 iterations in a row. The entire training process is performed off-line.

**Diversity Analysis.** The diversity of the regressors pre- and post-performing the CSEL Methods is depicted in Table 4 for all the datasets under estimation. Here diversity is defined as the average difference between the predictions made by the regressors. In the experiments we observed that the diversity drops rapidly when the accuracy goes up. At the point of convergence the regressors become very similar to each other and the ensemble of the results does not bring any more improvements at this point. As mentioned before diversity is an important condition to achieve a good performance. Apparently, simply distinguishing the teaching sets of the two regressors cannot help enough with maintaining the diversity and some strategies can be designed in this respect.

Moreover, when generating the multiple regressors with M$_h$ or M$_v$, there is a setback on their performance, i.e., $h_1$ and $h_2$ do not perform as good as the full model (Eq. 4) being trained on the full training set. For example, as shown in Table 3, $h_{1v}$ and $h_{2v}$ have a setback in accuracy compared to the full model. Although by simply assembling the two regressors together the accuracy can be brought back (see Table 1), some improvement could still be done in this respect to obtain an even better initial point at the beginning of the SSL process.

### 6.4 Qualitative Case Study

To have a more intuitive sense about how the algorithms work in different cases, we pick two users and two items from the dataset and look at their predictions. Let $u_1$ be an active user and $u_2$ be an inactive user, $i_1$ be a popular item and $i_2$ be an unpopular item.

Then four examples $x_{11}, x_{12}, x_{21}, x_{22}$ are drawn from the dataset with the given users and items, where $x_{ij}$ is given by $u_k$ to $i_j$. This way $x_{11}$ is rated by an active user to a popular item whereas $x_{22}$ is rated by an inactive user to an unpopular item, etc., and the performance of the algorithms on them can be observed. In this case study we pick the four examples that have the real ratings, $r_{11}, r_{12}, r_{21}, r_{22}$.

The results are given in Table 5, showing that providing accurate predictions also helps with personalization. That is, to recommend items that the end-user likes, but that are not generally popular, which can be regarded as the users’ niche tastes. By providing more accurate predictions for these items, the examples such as

### Table 5: Predictions made for the users and items with different popularity.

<table>
<thead>
<tr>
<th>Methods</th>
<th>$x_{11}$</th>
<th>$x_{12}$</th>
<th>$x_{21}$</th>
<th>$x_{22}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real Rating</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>FactCF</td>
<td>3.3124</td>
<td>3.2171</td>
<td>2.0155</td>
<td>3.7781</td>
</tr>
<tr>
<td>Context</td>
<td>3.3084</td>
<td>3.4605</td>
<td>1.9560</td>
<td>4.2217</td>
</tr>
<tr>
<td>Ensemble</td>
<td>3.2934</td>
<td>3.4913</td>
<td>1.8032</td>
<td>4.3433</td>
</tr>
<tr>
<td>CSEL$_h$</td>
<td>3.2313</td>
<td>3.4913</td>
<td>1.8351</td>
<td>4.241</td>
</tr>
<tr>
<td>CSEL$_v$</td>
<td>3.1971</td>
<td>3.7655</td>
<td>1.5456</td>
<td>4.671</td>
</tr>
</tbody>
</table>
$x_{12}$ or $x_{22}$ can be identified and recommended to the users who like them. Thus the users’ personalized tastes can be retrieved.

7. CONCLUSION

This paper resorts to the semi-supervised learning methods to solve the cold-start problem in recommender systems. Firstly, to compensate the lack of ratings, we propose to combine the contexts of users and items into the model. Secondly, a semi-supervised co-training framework is proposed to incorporate the unlabeled examples. In order to perform the co-training process and let the models teach each other, a method of constructing the teaching sets is introduced. The empirical results show that our strategy improves the overall system performance and makes a significant progress in solving the cold-start problem.

As future work, there are many things worth to try. For example, our framework provides a possibility for accommodating any kinds of regressors. One interesting problem is how to choose the right regressors for a specific recommendation task. Meanwhile, this semi-supervised strategy can be easily expanded to include more than two regressors. For example, three regressors can be constructed and a semi-supervised tri-training can then be performed on them. Finally, it is also intriguing to further consider rich social context [28] in the recommendation task.

Acknowledgements. This work is partially supported by National Natural Science Foundation of China (61103078, 61700994), National Basic Research Program of China (2012CB316006, 2014CB340500), NSFC (61222212), and Fundamental Research Funds for the Central Universities in China and Shanghai Leading Academic Discipline Project (B114). We also thank Prof. Zhihua Zhou for his valuable suggestions.

8. REFERENCES